# 莫比乌斯反演

<https://blog.csdn.net/acdreamers/article/details/8542292#comments>

莫比乌斯反演在数论中占有重要的地位，许多情况下能大大简化运算。那么我们先来认识莫比乌斯反演公式。

****定理：****IMG_256和IMG_257是定义在非负整数集合上的两个函数，并且满足条件IMG_258，那么我们得到结论

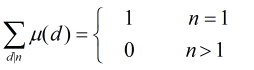
IMG_259

在上面的公式中有一个IMG_260函数，它的定义如下：

    （1）若IMG_261，那么IMG_262

    （2）若IMG_263，IMG_264均为互异素数，那么IMG_265

    （3）其它情况下IMG_266

对于IMG_267函数，它有如下的常见性质：

    （1）对任意正整数IMG_268有

    （2）对任意正整数IMG_270有

IMG_271

线性筛选求莫比乌斯反演函数代码。

void Init()

{

memset(vis,0,sizeof(vis));

mu[1] = 1;

cnt = 0;

for(int i=2; i<N; i++)

{

if(!vis[i])

{

prime[cnt++] = i;

mu[i] = -1;

}

for(int j=0; j<cnt&&i\*prime[j]<N; j++)

{

vis[i\*prime[j]] = 1;

if(i%prime[j]) mu[i\*prime[j]] = -mu[i];

else

{

mu[i\*prime[j]] = 0;

break;

}

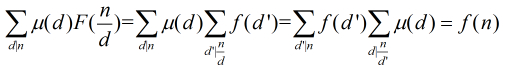
}

}

}

有了上面的知识，现在我们来证明莫比乌斯反演定理。

****证明****

********

证明完毕！

嗯，有了莫比乌斯反演，很多问题都可以简化了，接下来我们来看看莫比乌斯反演在数论中如何简化运算的。

****题目：**[http://bz.cdqzoi.com/JudgeOnline/problem.php?id=2818](http://bz.cdqzoi.com/JudgeOnline/problem.php?id=2818" \t "https://blog.csdn.net/acdreamers/article/details/_blank)**

****题意：****给一个正整数IMG_273，其中IMG_274，求使得IMG_275为质数的IMG_276的个数，IMG_277。

****分析：****对于本题，因为是使得IMG_278为质数，所以必然要枚举小于等于IMG_279的质数，那么对于每一个质数IMG_280，只

     需要求在区间IMG_281中，满足有序对IMG_282互质的对数。

     也就是说，现在问题转化为：在区间IMG_283中，存在多少个有序对使得IMG_284互质，这个问题就简单啦，因为

     是有序对，不妨设IMG_285，那么我们如果枚举每一个IMG_286，小于IMG_287有多少个IMG_288与IMG_289互素，这正是欧拉函数。所以

     我们可以递推法求欧拉函数，将得到的答案乘以2即可，但是这里乘以2后还有漏计算了的，那么有哪些呢？

     是IMG_290且为素数的情况，再加上就行了。

****代码：****

#include <iostream>

#include <string.h>

#include <stdio.h>

#include <bitset>

using namespace std;

typedef long long LL;

const int N = 10000010;

bitset<N> prime;

LL phi[N];

LL f[N];

int p[N];

int k;

void isprime()

{

k = 0;

prime.set();

for(int i=2; i<N; i++)

{

if(prime[i])

{

p[k++] = i;

for(int j=i+i; j<N; j+=i)

prime[j] = false;

}

}

}

void Init(){

for(int i=1; i<N; i++) phi[i] = i;

for(int i=2; i<N; i+=2) phi[i] >>= 1;

for(int i=3; i<N; i+=2)

{

if(phi[i] == i)

{

for(int j=i; j<N; j+=i)

phi[j] = phi[j] - phi[j] / i;

}

}

f[1] = 0;

for(int i=2;i<N;i++)

f[i] = f[i-1] + (phi[i]<<1);

}

LL Solve(int n)

{

LL ans = 0;

for(int i=0; i<k&&p[i]<=n; i++)

ans += 1 + f[n/p[i]];

return ans;

}

int main()

{

Init();

isprime();

int n;

scanf("%d",&n);

printf("%I64d\n",Solve(n));

return 0;

}

嗯，上题不算太难，普通的欧拉函数就可以搞定，接下来我们来看看它的升级版。

****题意：****给定两个数IMG_291和IMG_292，其中IMG_293，IMG_294，求IMG_295为质数的IMG_296有多少对？其中IMG_297和IMG_298的范围是IMG_299。

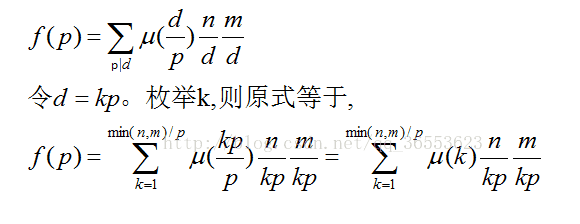
IMG_256

对于本题，我们设

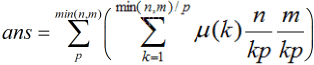
IMG_256 为满足IMG_256且IMG_256和IMG_256的IMG_256的对数

IMG_256为满足IMG_256且IMG_256和IMG_256的IMG_256的对数

      那么，很显然IMG_256，反演后得到IMG_256



因为题目要求是IMG_256为质数，那么我们枚举每一个质数IMG_256，然后得到



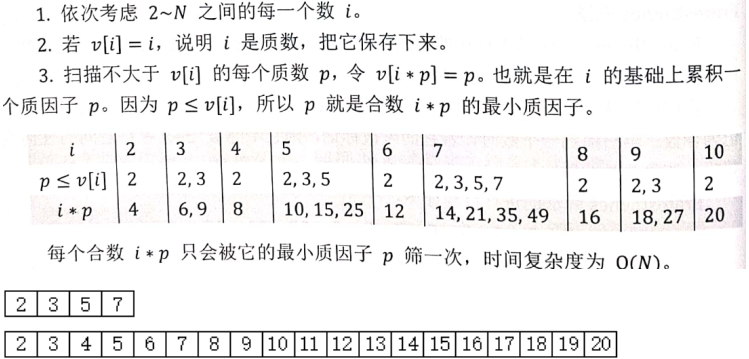
     如果直接这样做肯定TLE，那么我们必须优化。

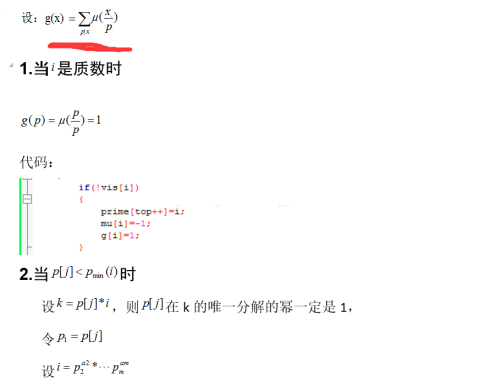
      我们设，那么继续得到IMG_256。

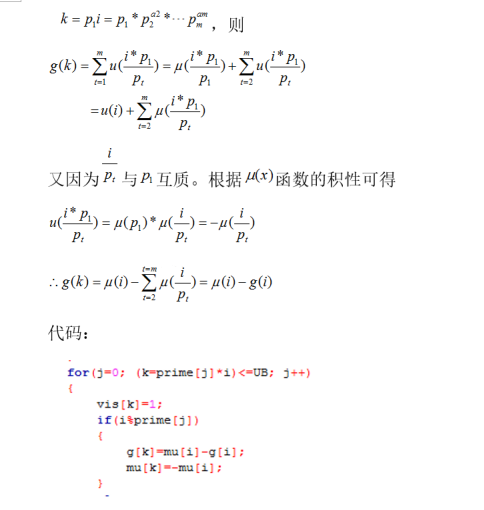
首先线性筛法的原理是枚举，小于等于的质数。

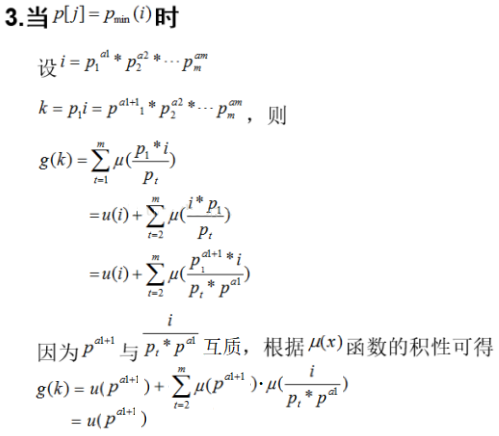
使得，即使得p[j]是k的最小质因数。 从而使得任何数只能由最小质因数转移过来，唯一了化转移方程,从而达到线性。

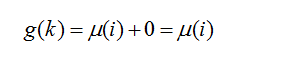
欧拉线性筛法











      （1）如果IMG_328整除IMG_329，那么得到



     （2）如果IMG_331不整除IMG_332，那么得到



#include <iostream>

#include <string.h>

#include <stdio.h>

using namespace std;

typedef long long LL;

const int N = 10000005;

bool vis[N];

int p[N];

int cnt;

int g[N],u[N],sum[N];

void Init()

{

memset(vis,0,sizeof(vis));

u[1] = 1;

cnt = 0;

for(int i=2;i<N;i++){

if(!vis[i]){

p[cnt++] = i; u[i] = -1;

g[i] = 1;

}

for(int j=0;j<cnt&&i\*p[j]<N;j++){

vis[i\*p[j]] = 1;

if(i%p[j])

{

u[i\*p[j]] = -u[i];

g[i\*p[j]] = u[i] - g[i];

}

else{

u[i\*p[j]] = 0;

g[i\*p[j]] = u[i];

break;

}

}

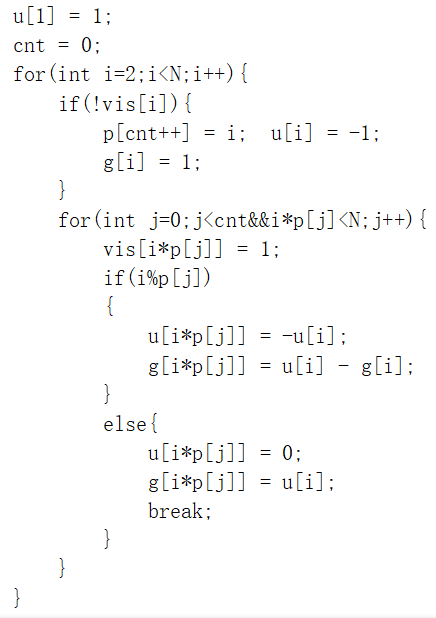
}

sum[0] = 0;

for(int i=1;i<N;i++)

sum[i] = sum[i-1] + g[i];

}

int main()

{

Init();

int T;

scanf("%d",&T);

IMG_320 while(T--)

{

LL n,m;

cin>>n>>m;

if(n>m) swap(n,m);

LL ans = 0;

for(int i=1,last;i<=n;i=last+1)

{

last = min(n/(n/i),m/(m/i));

ans += (n/i)\*(m/i)\*(sum[last]-sum[i-1]);

}

cout<<ans<<endl;

}

return 0;

}